

Appln No. 10/733,823

Amdt date August 5, 2005

Reply to Office action of July 14, 2005

REMARKS/ARGUMENTS

The above identified patent application has been amended and reconsideration and reexamination are hereby requested.

Claims 7 - 12 are now in the application. Claims 1 - 6 and 13 - 17 have been previously withdrawn in response to the Examiner requiring a restriction under 35 U.S.C. § 121, when the Applicant elected Claims 7 - 12 to be examined. Claim 7 has been amended. Claim 8 has been cancelled.

The Examiner has objected to certain informalities: The Examiner has objected to the term "uncorrected syndrome polynomial of the algebraic-coded message" used in Claim 7, asserting that is not clear how "an uncorrected syndrome polynomial" differs from the standard syndrome polynomial. Claim 7 has been amended to read "syndrome polynomial of the algebraic-coded message."

The Examiner has rejected Claims 7 - 12 under 35 U.S.C. § 112 for failing to comply with the enablement requirement, asserting that use of the terms "uncorrected syndrome polynomial" and "corrected syndrome polynomial" in Claim 7 is not supported in the specification. Claim 7 has been amended and does now not include either of the terms "uncorrected syndrome polynomial" or "corrected syndrome polynomial."

The Examiner has rejected Claims 7 - 12 under 35 U.S.C. § 112 as being indefinite. The Examiner has rejected Claim 7, which calls for "a plurality of polynomial storage devices being adapted to store polynomials," and upon which Claims 9 - 12 depend, asserting that polynomials are abstract mathematical elements and that is not clear how an abstract mathematical

Appln No. 10/733,823

Amdt date August 5, 2005

Reply to Office action of July 14, 2005

element can be stored or how they are stored in a hardware memory device. The Examiner has also rejected Claim 9, which calls for "the inversionless calculator temporarily stores the discrepancy values in a first discrepancy variable and a second discrepancy variable, storing in the second discrepancy variable temporarily the last value previously stored in the first discrepancy variable," asserting that variables are not storage devices and it is not clear how a value can be stored in a variable.

However, one reasonably skilled in the art will appreciate that in a plurality of programming languages, often referred to as assignment-based languages, variables are storage devices. For example, the Free On-Line Dictionary of Computing [FOLDOC], a service of Imperial College London (available online at <http://wombat.doc.ic.ac.uk/foldoc>) defines "variable" as:

<programming> (Sometimes "var" /veɪr/ or /var/) A
named memory location in which a program can store
intermediate results and from which it can read . . .
them. Each programming language has different rules
about how variables can be named, typed, and used.
Typically, a value is "assigned" to a variable in an
assignment statement. . . .

(underlining added for emphasis)

See also McGraw-Hill Encyclopedia of Science & Technology (1998) (defining "variable" as "[computer science] A data item, or specific area in main memory, that can assume any of a set of values."). Assignment-based languages include such popular programming languages as ALGOL, BASIC, COBOL, FORTRAN, PASCAL,

Appln No. 10/733,823

Amdt date August 5, 2005

Reply to Office action of July 14, 2005

MODULA, C, C++ and JAVA (for comparison: non-assignment based languages are generally referred to as functional languages and include, e.g., LISP and PROLOG). See FOLDOC (defining "assignment" as "[s]toring the value of an expression in a variable. This is commonly written in the form "v = e". In Algol the assignment operator was "[:=" (pronounced "becomes") to avoid mathematicians['] qualms about writing statements like x = x+1. Assignment is not allowed in functional languages"); see also McGraw-Hill (defining "assignment statement" as "[computer science] A statement in a computer program that assigns a value to a variable."). In fact, on pages 8 - 9 of the specification of the present application, part of the code of an exemplary embodiment of the present invention is illustrated using PASCAL notation. See page 8, lines 21 - 22.

Referring to this illustrative code example in the specification, the shown code represents a procedure called "FindLocatorBMC." Page 8, line 24. At the very beginning of the procedure's code, variables are defined, as indicated by the keyword "VAR." Page 8, line 26. These definitions will normally cause a computer to allot space in its (hardware) memory in order to store values in connection with the defined variables. See definitions of "variable," supra. Values are then stored in these variables throughout the procedure, as indicated by the uses of the assignment operator "[:=". Page 9, lines 5 -13, 22-23, 28 - 29, 32 - 33, 37 39, 41, 44 - 45. In particular, the variables used to store discrepancy values in this code example are "Del" and "Del0." "Del0" is initially assigned the value 1, i.e. the integer number 1 is stored in the

Appln No. 10/733,823

Amdt date August 5, 2005

Reply to Office action of July 14, 2005

computer's hardware memory device in the space allotted for the variable "Del0." Page 9, line 7. Similarly, "Del" is initially assigned the value 0. Line 12.

Furthermore, variables may not only be used to store simple numbers. Variables representing data structures can be used to store complex mathematical elements, including polynomials. Referring again to the code example, the variables "B" and "TempPoly" are instances of a data structure of type "Polynomial" and are local variables instantiated within the procedure. Page 9, lines 1 - 2. In addition, two variables named "Syndrome" and "Locator," also of the type "Polynomial" are provided to the procedure "FindLocatorBMC" as parameters. Page 8, line 24. The data structure type "Polynomial" will generally be defined somewhere in the code environment of the procedure "FindLocatorBMC."

A data structure for storing polynomials can, for example, be implemented via an array of numbers, with each number corresponding to a coefficient with different power, e.g., an array $[a_0 a_1 a_2 a_3 a_4 a_5]$ could represent the polynomial

$$a_0^0 + a_1^1 + a_2^2 + a_3^3 + a_4^4 + a_5^5.$$

Other data structures can be used to store polynomials, such as linked lists. In modern, object-oriented languages (such as Smalltalk, C++ or Java) definitions of data types include not only the structure (often referred to as attributes), but also operations that can be performed upon the data type (generally referred to as methods). Such data definitions are known as classes; an instance of such a class is known as an object. See FOLDOC (defining "class" as

Appln No. 10/733,823

Amdt date August 5, 2005

Reply to Office action of July 14, 2005

"<programming> The prototype for an object in an object-oriented language; analogous to a derived type in a procedural language. A class may also be considered to be a set of objects which share a common structure and behaviour. The structure of a class is determined by the class variables which represent the state of an object of that class and the behaviour is given by a set of methods associated with the class. . . .") For example, one can define a class "Polynomial," which stores polynomials in arrays, and include operations such as "add" or "multiply" that can be performed upon a polynomial stored in the array.

The Applicant submits that the above examples only serve to illustrate how polynomials can be stored and how variables can be used as storage devices generally, respectively, but are not meant to limit interpretation of the Claims at issue to these examples or in any other way.

The Applicant further submits that in light of the amendments and explanations above, Claim 7 is supported by the specification and neither Claim 7 nor Claim 9 is indefinite or fails to add meaningful limitations. Therefore, Claims 7 and 9 - 12 are now in compliance with 35 U.S.C. § 112.

The Examiner has rejected Claims 7, 9, 11 and 12 under 35 U.S.C. § 103(a) as being unpatenable over U.S. Patent No. 6,209,115 to Truong et al. in view of U.S. Patent No. 5,446,743 to Zook. The Applicant has requested amendment of Claim 7 to further include the following elements:

a binary state storage device being adapted to
store a binary state, operably connected to the

Appln No. 10/733,823

Amdt date August 5, 2005

Reply to Office action of July 14, 2005

inversionless calculator and the arithmetic-logic components; and

an uncorrectable error indicator, operably connected to the inversionless calculator and the arithmetic-logic components, the inversionless calculator iterating through the location of errors in the algebraic-coded message, while:

determining the existence of errors, the location and magnitude of the errors and the discrepancy values in the algebraic-coded message,

storing a state variable in the binary state storage device before a first iteration to indicate that no uncorrectable error has been detected and iterating through the errors while updating the state variable to indicate whether an uncorrectable error has been detected,

using the uncorrectable error indicator to indicate that the algebraic-coded message is uncorrectable if the state variable contains an indication that an uncorrectable error has been detected after a final iteration; and

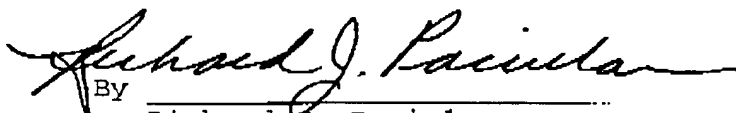
using the uncorrectable error indicator to indicate that the algebraic-coded message is not uncorrectable if the existence of no errors has been determined or the state variable contains an indication that no uncorrectable error has been detected after the final iteration.

AppIn No. 10/733,823
Amdt date August 5, 2005
Reply to Office action of July 14, 2005

Neither Truong et al. nor Zook teach uncorrectable error indication. Therefore, the Applicant submits that Claim 7, as amended, is not unpatenable over Truong et al. in view of Zook. Furthermore, Claims 9 - 12 depend on Claim 7. As such, Claims 9 - 12 are also believed allowable based on Claim 7.

Accordingly, in view of the above amendment and remarks, it is submitted that the Claims are patentably distinct over the prior art and that all objections and rejections to the Claims have been overcome. Reconsideration and reexamination of the above Application is requested.

Respectfully submitted,
CHRISTIE, PARKER & HALE, LLP


By Richard J. Paciulan
Reg. No. 28,248
626/795-9900

RJP/mac
MAC PAS636276.1 * 08/5/05 11:11 AM